Table of Contents

❑ INTRODUCTION

In this project I’ll be implementing a single layer perceptron from scratch to classify handwritten digits. The handwritten digits come from MNIST dataset which consists of 60000 images along with their labels. Each image is 28*28 pixels wide and is stored in numpy array format. A perceptron is also known as a shallow neural network due to lack of any hidden layer. It consists of one input and one output layer. Although it is one of the most popular for classification problems it can also be applied to regression tasks.

❑ METHODOLOGY

➔ The images are stored as (60000, 28, 28) tensors. For this purpose I’ll flatten all the images into (60000, 28*28) matrix. The perceptron takes an entire image (flatten 784 pixels) as one training example at a time.

➔ After that I’ll normalize all images by dividing each one of them by 255.0. For task 1 and 2 I’ll work with binary images and for task 3 I’ll work with original images which are in grayscale format.

➔ To convert images in binary form I’ll pick a threshold value between 0 and 255 and filter values in the image arrays, keeping only values above the threshold and replacing remaining values with 0. This will reduce the pixels in the image further decreasing the clarity and detail.

➔ For training I’ll pick a learning parameter. Choice of learning parameter will affect the performance of the model. Also I’ll choose weights and biases which will learn from the training.

➔ I'll run each model for 10000 iterations and in each iteration I'll perform two tasks: forward propagation and backward propagation.

➔ In forward propagation I'll first compute hypothesis function, softmax activation and cost function. In backward propagation I'll use cost and to optimize weight by calculating gradients and then updating weights and biases using those gradients.

➔ Accuracy is the total number of correct predictions divided by the total number of predictions.

❑ EQUATIONS USED:

1. Forward Propagation:

Let X be an $i \times j$ matrix containing i training images and j pixels in each training image. Let W and b be the matrix of weights and bias respectively.

The hypothesis function $h(x)$ is given by the equation:

$$h(x) = X.W + b$$

The activation function and output layer is a softmax function. Let S denote a softmax function.

$$S(z) = \frac{e^z}{\sum e^z}$$

For our case: $y\_pred = S(h(x_{ij})) = \frac{e^{h(x_{ij})}}{\sum_j e^{h(x_{ij})}}$

Where y_pred is the matrix of $i \times k$ predictions. In our case k is 10 for 10 classes to be predicted.

2. Loss Function:

Let $L(y, y\_pred)$ be the loss for one training example.

$$L(y, y\_pred) = -y * log(y\_pred)$$

Cost for entire training is given by

$$C = \frac{\sum_k L(y_k, ypred_k)}{i}$$

3. Backward propagation:

C is the cost of the training, X is the training set and W and b are weights matrix and bias matrix. Let ∝ be the learning parameter.

$$\frac{\partial C}{\partial W} = \frac{\partial C}{\partial y\_pred} \times \frac{\partial y\_pred}{\partial X} \times \frac{\partial X}{\partial W}$$

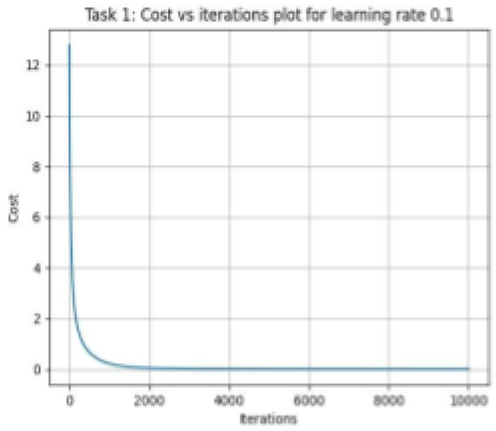
$$\frac{\partial C}{\partial b} = \frac{\partial C}{\partial y\_pred} \times \frac{\partial y\_pred}{\partial X} \times \frac{\partial X}{\partial b}$$

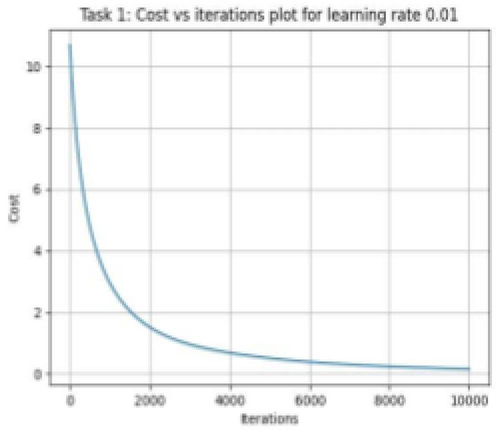
Update weights and biases

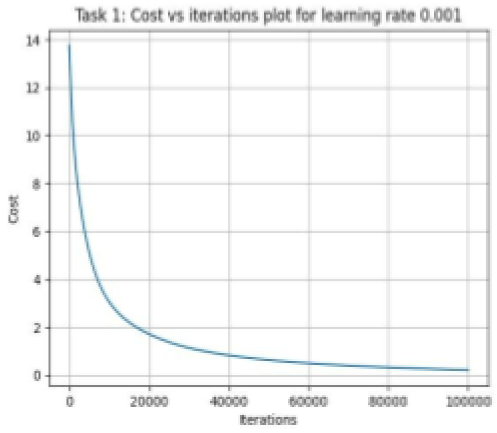
$$W := W - \propto \frac{\partial C}{\partial W} ; b := b - \propto \frac{\partial C}{\partial b}$$

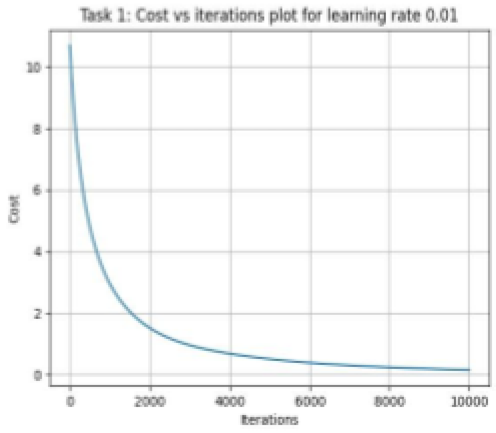
❑ RESULTS:

---------------------------------- TASK 1 ---------------------------------



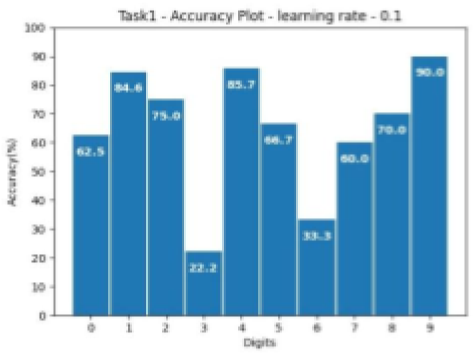


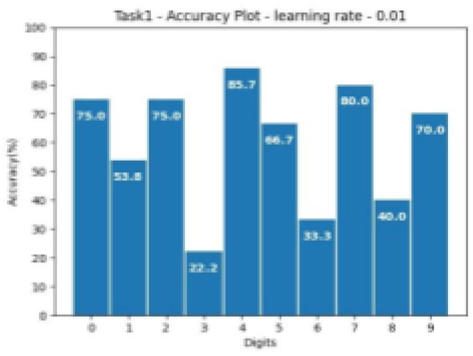


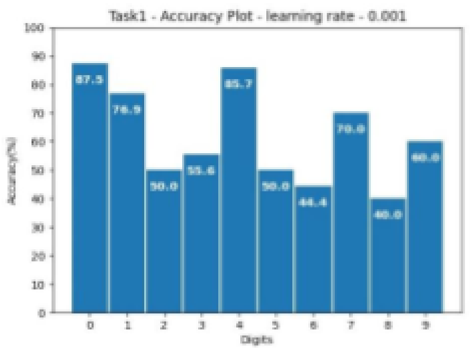


Learning Rate	Iterations	Accuracy	Loss
0.1	10000	0.65	0.005
0.01	10000	0.59	0.235
0.001	10000	0.46	2.93
0.001	100000	0.64	0.222

- ➔ From the graphs and the table it is observed that 0.1 is the best value of learning parameter as it minimises the training loss and gives maximum accuracy of 65%.
- ➔ The graph also shows two iterations with learning rate 0.001. First one is 10000 Iterations, as it is clear that the model could not find the minimum when the value of the learning rate is so small. Second one is 100000 iterations. After running for these many iterations the model was able to converge to a loss of 0.222 with an accuracy of 64%.

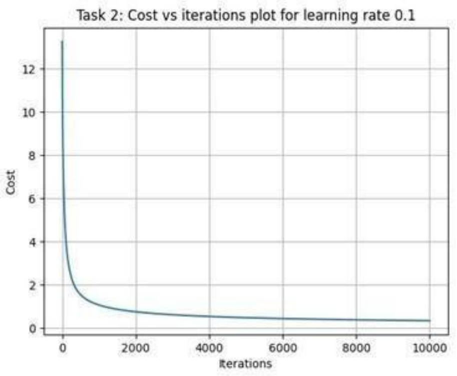


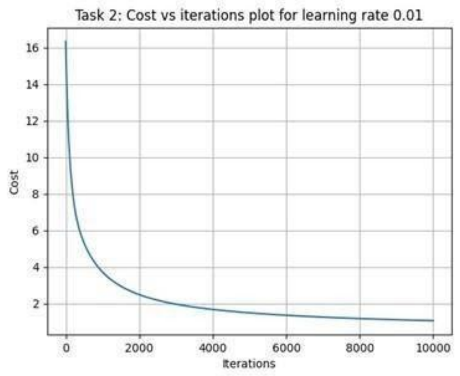


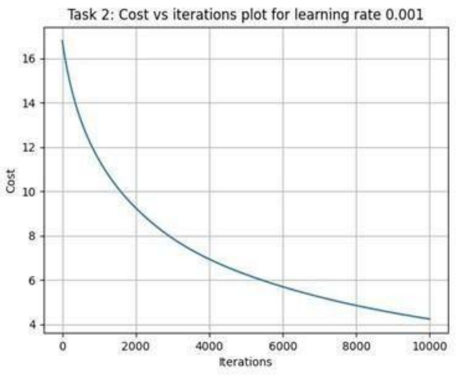


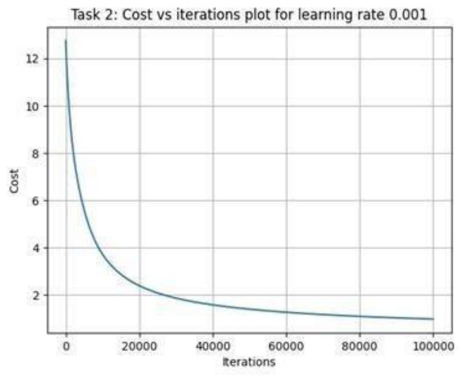
- ➔ From the above graph it can be concluded that the model failed to classify digit 3 in the first two cases when learning parameters are 0.1 and 0.01, but is recognised when the value is 0.001. In this case lowering the value of learning rate yielded a better classifier. This is also confirmed from the above table where accuracy is 64%.

------------------------------------ TASK 2 ----------------------------------





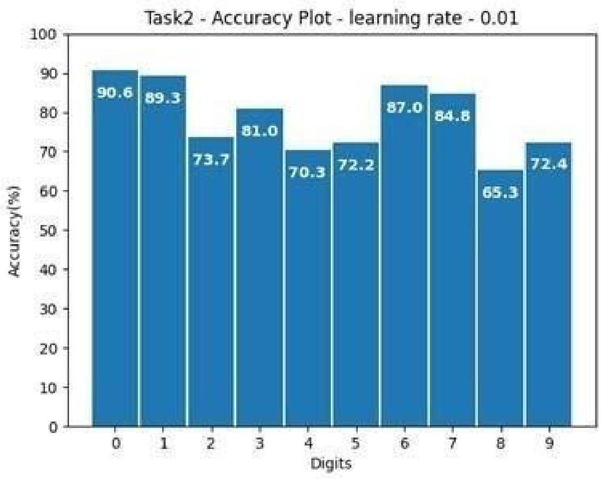


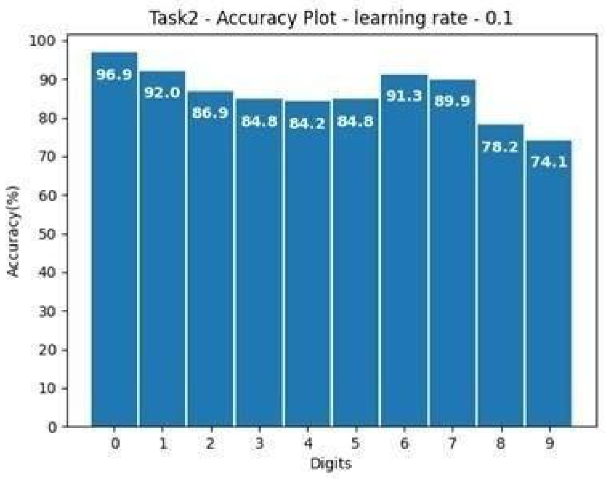


Learning Rate	**Iterations**	**Accuracy**	**Loss**
0.1	10000	0.854	0.347
0.01	10000	0.784	1.057
0.001	10000	0.399	4.229
0.001	100000	0.771	0.987

➔ The graphs show that the model takes more iterations to converge to minima as compared to previous graphs. This is due to the fact that since the value of the learning rate is small it results in very little change in the existing values of weights and biases leading to slow training.

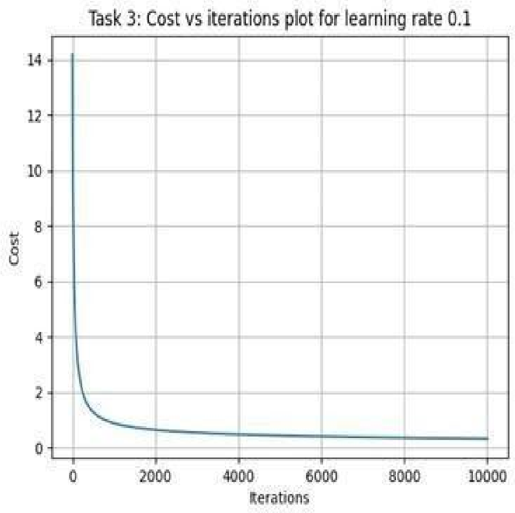
- ➔ Like the previous model this model also failed to converge to minimum within 10000 iterations when the value of the learning rate is 0.001
- ➔ The graph also shows two iterations with learning rate 0.001.
- ➔ First one is 10000 Iterations, as it is clear that the model could not find a minimum when the value of learning rate is so small.
- ➔ Second one is 100000 iterations. After running for these many iterations the model was able to converge to a loss of 0.222 with an accuracy of 64%.

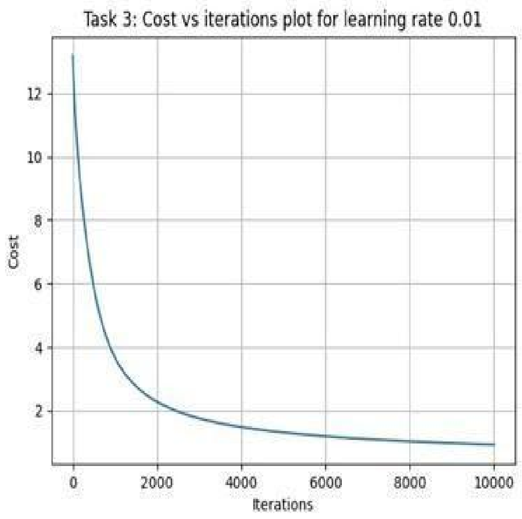


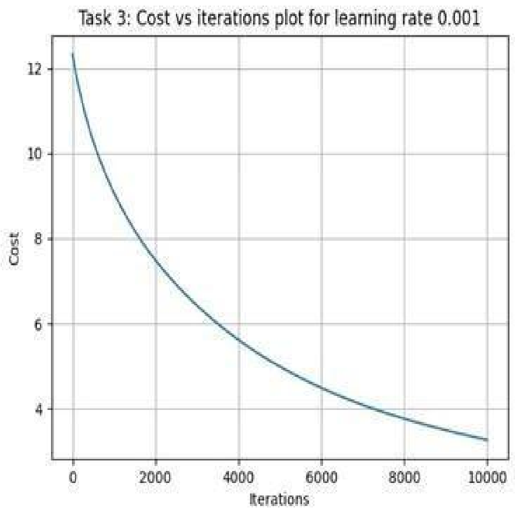


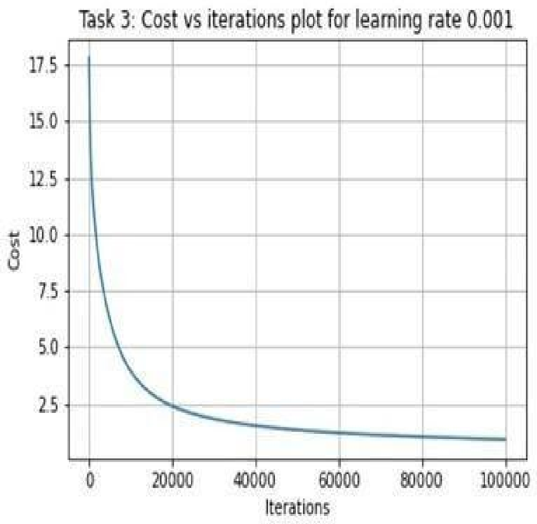
- ➔ From the graph it is seen that accuracy for each digit is more when learning rate is 0.1 and overall accuracy is also 85%. The accuracy decreased slightly when the learning rate was changed from 0.1 to 0.01. This further decreases the model accuracy to 78.4%.

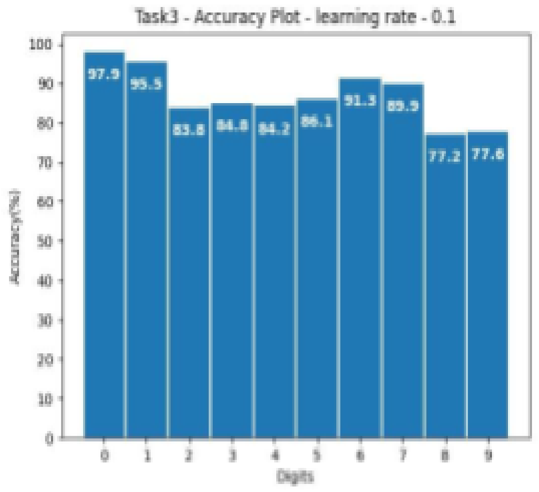
------------------------------------ TASK 3 ----------------------------------

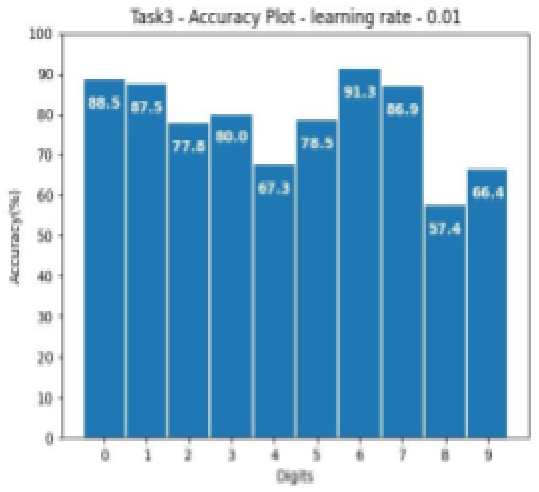

Task 3: Cost vs iterations plot for learning rate 0.1
Cost
Iterations
0
2
4
6
8
10
12
14
0
2000
4000
6000
8000
10000

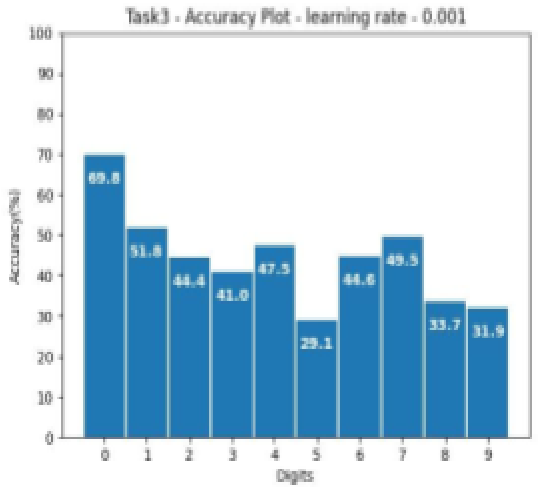

Task 3: Cost vs iterations plot for learning rate 0.01
Cost
Iterations
2
4
6
8
10
12
0
2000
4000
6000
8000
10000


Task 3: Cost vs iterations plot for learning rate 0.001
Cost
Iterations
4
6
8
10
12
0
2000
4000
6000
8000
10000


Task 3: Cost vs iterations plot for learning rate 0.001
Cost
Iterations
2.5
5.0
7.5
10.0
12.5
15.0
17.5
0
20000
40000
60000
80000
100000







Learning Rate	**Iterations**	**Accuracy**	**Loss**
0.1	10000	0.867	0.3215
0.01	10000	0.779	0.923
0.001	10000	0.61	0.1644
0.001	100000	0.792	0.9189

----------------------------------- TASK 4 ---------------------------------

Comparing SVM and Single Layer Perceptron Results:

- ➔ In the SVM model, the maximum accuracy was achieved using RBF kernel which was about 96%. The maximum accuracy achieved in case of Single Layer Perceptron is 86% with a learning rate of 0.1.
- ➔ In case of binary images using a threshold value of 120, the maximum accuracy obtained in case of SVM model is 89% and the maximum accuracy obtained in case of perceptron is 86%
- ➔ Due to the fact that perceptron models require higher number of iterations to converge to minimum value, this model consumes more time than SVM models.

❑ CONCLUSION:

In conclusion, the learning rate is a crucial deciding parameter for a perceptron's performance. For lower values of learning rate the model has to be run 10 times more than the models using larger values of learning rate. Second the performance of the models also depends upon the size of data. More number of examples help reduce under fitting and achieve good accuracy. Finally SVM and perceptron are equally good classifiers when compared based on accuracy. Perceptron is slower to train in comparison to the SVM.

❑ REFERENCES:

https://www.tutorialspoint.com/tensorflow/tensorflow_single_layer_perceptron.htm

http://www.cs.stir.ac.uk/courses/ITNP4B/lectures/kms/2-Perceptrons.pdf

https://towardsdatascience.com/single-layer-perceptron-in-pharo-5b13246a041d

https://medium.com/@michaeldelsole/a-single-layer-artificial-neural-network-in-20-lines-of-python-ae34b47e5fef